Craggs, B. (2019). A Just Culture is Fundamental: Extending Security Ergonomics by Design. In *2019 IEEE/ACM 5th International Workshop on Software Engineering for Smart Cyber-Physical Systems (SEsCPS)* Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/SEsCPS.2019.00015

A Just Culture is Fundamental: Extending Security Ergonomics by Design

Barnaby Craggs
Department of Computer Science, University of Bristol, UK
Email: barney.craggs@bristol.ac.uk

***Abstract*—Human error when developing and using smart cyber physical systems is inevitable. Earlier work has set out Security Ergonomics by Design—principles by which developers of systems can ensure that the active user error cannot occur when latent system failures introduced in development are in play. This paper underpins these principles by showing there is a fundamental need to adopt a Just Culture within which i) user error is captured for improvement in the development cycle, and ii) to provide software engineers assurance that their own mistakes are not automatically punished but rather treated as learnings that can be fed back into building safer and more secure practice.**

***Index Terms*—Security Ergonomics, Just Culture, Human Factors, Internet of Things, Cyber-Physical Systems**

I. Introduction

Nascent technologies such as connected medical devices, smart city sensor networks, autonomous vehicles, smart home automation and connected toys present a number of design and development challenges to ensure they remain both safe and secure. Any single class of device can be a complicated mix of software, hardware, human behaviour, environmental influence, and cultural practice. Together, colloquially referred to as the Internet of Things (IoT), these devices often occupy the same physical spaces, are used by the same people, interact and exchange information with each other. The IoT as a collection of devices is a fantastically complicated system of systems.

In prior work Craggs and Rashid [1] recognised that flawed design choices (latent failures) can lead to, and sometimes make inevitable, human error in using IoT and other cyber physical systems (CPSs)—this error resulting in devices that at best become insecure and at worst become unsafe. By way of mitigation, Craggs and Rashid drew upon methodology and practice from other safety-concerned domains and proposed *Security Ergonomics by Design* (SEbD), in which five key design principles for smart cyber physical systems guided developers in ensuring that:

- **Proactive Design**. The design of CPSs is proactive in identifying existing and known latent failures and active human error before they happen, rather than providing a reactive response to events as they occur.
- **Embedded Nature**. The approach of security ergonomics embeds directly into the design and architecture of CPSs, rather than being applied retrospectively.
- **Encouragement of Pro-Secure Behaviour**. Non-erroneous secure user behaviour is, by default, encouraged and possibly even rewarded. In circumstances where it is critical to the stability of systems as a whole such behaviour should be enforced.
- **Non-Alignment**. Active user error and developer introduced latent failure does not occur concurrently.
- **External Validation**. Security ergonomic based design is introspective, recognising that designers and developers themselves are human and prone to assumption, bias and error. Best practice software engineering techniques to remove that error should be employed, designs should be validated by third-parties.

This paper continues and extends SEbD by addressing its lack of method for the identification of existing latent failure and active error, essential to enable *Proactive Design* and therefore the whole principle set. Such a method can be drawn from aviation and medical practice where this understanding is critical to the re-design of previously failed and un-safe systems. This method is:

- **Just Culture**. A safe environment within which security incidents are captured, evaluated and post mortemed, without automatic blame, in order to provide detail into the security ergonomics approach for the design of safe and secure cyber-physical systems.

II. Just Cultures

"Incidents are unplanned investments" Allspaw. 2018

Stemming from safety-aware domains, Just Culture [2] is "*a culture of trust, learning and accountability,*" [3] which provides a safe environment, after an incident has occurred, within which interactions between aspects of a CPS—the software, hardware, liveware (humans), culture and environment (SCHEL) [4]—can be understood, and used to feedback into systemwide improvement. Just Culture [2] can be viewed as "*necessary for operating safely*" [5] by allowing a kaizen approach to iterative and continuous system improvement towards removing human error [6].

A. Blame Cultures Are Not Helpful

History is filled with examples where people have erred, either accidentally or deliberately. In 2016, thousands of ordinary users unwittingly bought and configured cheap, insecure web-cameras using the simple usable security enhancing technique of wireless protected setup (WPS) on their home routers. In that simple process many forgot to change the default administration

credentials paving the way for the Mirai botnet to undertake distributed denial of service attacks and reek global havoc [7]. In 2019 we are still seeing Mirai variants propagate on the Internet.

In May 2017, the ransomware WannaCry targeted Microsoft Windows XP based computers around the globe. In the United Kingdom, the National Health Service (NHS) was crippled [8], surgical operations were cancelled and the attack was estimated to cost in excess of £92M. The NHS was not a specific target for the attack, but an over reliance upon key systems running Windows XP without national support contracts being in place, and an organisational culture not attuned to the vulnerability of critical services from cyber attack meant it was severely impacted. In the days and weeks after WannaCry surfaced, much attention was paid as whom was to blame for the attack. Attribution was paid to nation-states, the National Security Agency (NSA) who had developed the initial underlying Windows exploit, Microsoft for not providing critical patches to operating systems in-perpetuity for free, the NHS for their lack of response, the United Kingdom's government for not funding needed support contracts, training and staff. And by no means least, the users of those NHS systems who had effectively spread the worm, and failed to take immediate action to halt its spread.

This is a long list of potentially culpable parties and, as Nietzsche [9] points out, the need to find a cause, or person/party, responsible is somewhat fundamental to human nature as to not do so implies a loss of control and is distressing. The reality is they all played a role but the initial race to assign blame did nothing to help understand and dissect the problem, nor to ensure it could not happen again. The reason being that when blame is applied to error, especially very public potentially life-threatening error, all parties are keen to separate themselves from being held accountable. And when the users, owners, developers and maintainers of a system all actively seek to avoid what might be a punitive situation, the facts of what and how something occurred become lost in obsfucation.

B. The Benefits of Being Just

Dekker [10] states how maintaining a Just Culture benefits all system stakeholders:

1) For the organisation itself, without a Just Culture it is all but impossible to truly know what is going on.
2) For those who work in the organisation, the incentive is being free to concentrate on doing a quality job rather than on limiting personal liability. Research [11] has shown that not implementing a Just Culture is detrimental to morale, commitment, job satisfaction and willingness to step outside their role.
3) For users, a Just Culture de-prioritises short-term liability minimising measures and steers organisations towards long-term investment in safe and secure products.

C. In Practice

Mature safety aware industries, such as aviation, have recognised these benefits and developed Human Factors / Ergonomics (HF/E) practice to help people avoid the dissonance between the potential and automatic blame that traditionally would have been applied even when doing the correct thing and admitting they have erred. Without that insight it would be impossible to re-design systems to perform better and with less error.

In software development: One of the first software development-based companies to adopt Just Culture was the online marketplace for handmade & vintage items, Etsy.com. In 2012 John Allspaw, the then senior vice president of Etsy.com's infrastructure and operations team, wrote a blog post [12] as to how they had discovered and were applying "*blameless postmortems*" to their services development programme as a technique for reducing software engineer-led error. Etsy.com's intent was to allow their software engineers, whose actions had contributed to an error, to provide details about: i) the actions they had taken and when, ii) the effects and impacts they had personally observed, and iii) their own understanding of events in chronological order.

Importantly Etsy.com had recognised that for software engineers to willingly self-report they had to trust that by doing so they would not automatically face some form of punishment or retribution.

In CPS cyber security: In the domain of cyber security limited attention has been paid to the application of HF/E methodology for safe and secure systems, e.g., [1], [13], [14], [15], [16]. Just Culture has been discussed by HF/E and security practitioners in online forums, e.g., [17], [18]. Hayashi [19] lays claim to the need for Just Culture when looking at schemes for securing information systems, but focusses exclusively upon identifying aspects of data processing more from a pre-emptive policy perspective rather than as an approach to understanding security events. At the time of writing, no academic work has looked to the need for Just Culture as a fundamental requirement for the development of safe and secure cyber physical systems.

III. Just Culture as the First Principle

A. Enabling Security Ergonomics

To this point we have discussed Just Culture as being a process for dissecting events for the purposes of understanding what occurred and how this can be used for system improvement. Within this context, Just Culture is an enabling process, as it allows a space for security-related events to be safely collated and to feed into the second principle of proactive design.

Within SEbD, Just Culture plays an important second role in recognising and capturing another form of human error - that of the developer themselves, ala Etsy.com. Research in the safety domain has shown that, whilst technological advances can remove cognitive load from humans, and to a degree mitigate user error, there comes a point where this "*simply shifts the error*" [20] to those developing the system. Software engineers are also prone to error and so it is vital for safe and secure smart CPS that this error, as best as possible, is removed from the development cycle. As such, and alongside

the principle of external validation, Just Culture should also be practised within, and upon, the design and development teams themselves.

Given the dual pivotal role within SEbD, this paper proposes that Just Culture prepends the existing five principles, and becomes the new first principle of Security Ergonomics by Design, thusly: **1**) Just Culture, **2**) Proactive Design, **3**) Embedded Nature, **4**) Encouragement of Pro-Secure Behaviours, **5**) Non-Alignement, and **6**) External Validation.

B. Considerations for Defining Just Culture

A Just Culture provides people with confidence that by reporting errors, in confidence, they will be fairly investigated and result in visible and positive change. This is not to say accountability (or blame) is not a valid outcome—where deliberate subversion of systems and processes occurs culpability can lay with people. As with the prior set of principles, application is not meant to be prescriptive of method—for example with respect to external validation, the principle does not dictate method but rather calls upon best practice as it pertains to the environment within which the system is designed to reside. The same is true of Just Culture. What works for one development team as a process for capturing and dissecting developer error, may not fit with another team.

To achieve a culture of trust, one in which people will report errors, accidents and near-misses, the processes by which the report is made, handled, discussed and acted upon needs to be defined and clearly signposted for people to use. Crucially to establish any *just* process three questions should be asked: *Firstly*, who gets to draw the line between acceptable and unacceptable behaviour? *Secondly*, what and where should the role of domain expertise be in setting that line? And, *thirdly*, how well insulated are internal Just Culture processes and data.

Dekker [10] suggests that in relation to who sets the line, the clearer this responsibility is conveyed "*practitioners [software engineers] will suffer less anxiety and uncertainty... in the wake of an occurrence.*" Care should be taken with reliance upon domain expertise alone, whilst experts are better able to empathise and understand context and possibly be able to better avoid hindsight bias, they can be subject to other bias in line setting—for example by admitting that error is inherent in their activity rather than personal failure is to see themselves as equally vulnerable.

C. Application of Culpability

People err, both in developing and using smart systems. This is inevitable at some point and SEbD guides software engineers in how to avoid allowing this user (active error) and developer error (resulting in latent failure) to coincide. In applying Just Culture to the practice of software engineering, Reason's Culpability Decision Tree [21] (see figure 1) provides a baseline for understanding and agreeing how developer error can be assessed. The model, whilst generic and having been developed initially for aviation post mortems, looks to five areas of inquiry each of which ascribes diminishing *blame* or culpability to the person who's actions are in question.

Intention: Deliberate acts, even safety and security related ones, are not automatically bad. The intention behind the act is more important when understanding what happened and, to what extent someone is culpable. Within software engineering it is entirely possible, for example, that an error filled code library is used with the intention of cryptographically signing the communications between smart things. The intent was wholesome. Where an engineer had deliberately altered the code to break the cryptography the intent was subversive and could be seen as a sabotage rendering the engineer culpable for the latent failure of the system. Proactive Design & External Validation (*P2 & P6* of SEbD) provide guidance to firstly expect potentially poor library code as a risk, and secondly to utilise more than one person/method to check resultant code prior to release.

Impairment: Following the same example, impairment simply asks whether the engineer was using what would in their domain be considered an unauthorised substance—one considered to impair their cognitive ability to detect that erroneous code. Domains will have differing viewpoints on impairment. However pressures on software engineers (for example, commercial or timeliness) leads to longer and harder working practices, and in turn, substance abuse [22]. This being clearly linked to cognitive impairment [23]. Pro-Secure Behaviour *(P4)* would be in play here as, clearly, being impaired is not pro-secure. This may be an instance of where mandatory substance testing might be an enforced pro-secure behaviour.

Violation: Humans routinely deviate from what procedure considered safe (or secure), e.g., reusing simple passwords for convenience. The inquiry here revolves around whether the procedures themselves were not only available to the engineer, but also workable. Using our example, procedure that requires peer validation of security code libraries before implementation may be documented, but if staffing pressures meant peers were unavailable and self-acceptance of code was the norm, then clearly the system (the organisation's project management in this case) plays a role and diminishes personal culpability. Just Culture (*P1*) is the primary mitigating principle, providing the mechanism by which a violation can be safely reported. Knowing where software engineers might deviate from procedure and feeding this into evolving procedure through Proactive Design (*P2*), and Non-Alignment (*P5*) is crucial in ensuring engineers do not violate (a form of active error) when a latent failure in the procedures is known.

Substitution: Directly related to violation is the question of whether any other engineer would, or could, have made the same error. Clearly if peers were unavailable the system was failing. But what training or skills should the engineer have had and brought to bear? Where they personally negligent in self-accepting or was that a result of a lack of training/experience? Could the same outcome have occurred if another engineer had been in place. Applying External Validation (*P6*) practice to development can help to resolve substitution issues. If techniques like pair-coding sees both engineers apply the same error then patently substitution is applicable.

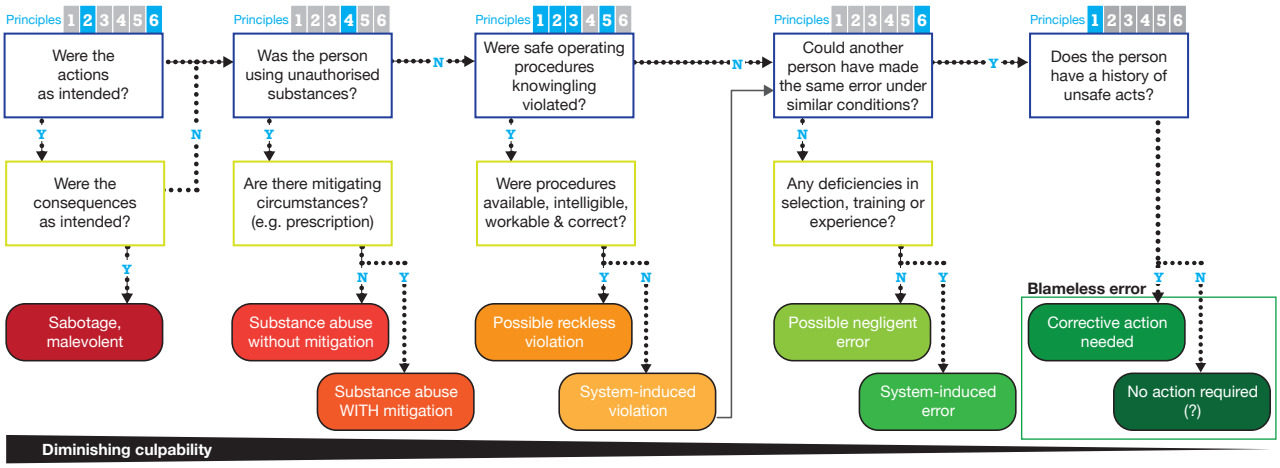

Fig. 1: Reason's *culpability decision tree*, with SEbD principles as they relate to inquiry areas.

History: The final area looks to whether the engineer has a history of similar behaviour. In software engineering terms this is harder to conceptualise, however one example would be someone who repeatedly self-accepts based on a lack of experience and requires corrective action perhaps through training or peer mentoring. A Just Culture (*P1*) should maintain clear and confidential documentation of security related events—both to be used within the improvement cycle but also to facilitate the final historical line of inquiry should it be needed.

IV. Conclusion

Prior work presented Security Ergonomics by Design—a set of principles to enable software engineers to develop safe and secure smart cyber physical systems. This paper corrects an omission of the fundamental need for a Just Culture in underpinning these principles. Drawing on lessons and practice from more mature safety domains, Just Culture enables the capture of user error, and for engineers to share their own development errors in a trusting environment which is designed not to apportion blame but rather provide iterative systemwide improvement based upon those failings.

Given the lack of research around the use of Just Culture in security-related event reporting, future work is planned to examine both the prevalence and use of Just Culture, and in turn to derive domain-specific decision trees for event post mortems and best practice for implementation within organisations developing smart cyber physical systems.

Acknowledgment

This work has been funded by the UK EPSRC as part of the PETRAS IoT Research Hub - Cybersecurity of the Internet of Things grant no EP/N023234/1.

References

[1] B. Craggs and A. Rashid, "Smart cyber-physical systems: beyond usable security to security ergonomics by design," in *Proceedings of the 3rd International Workshop on Software Engineering for Smart Cyber-Physical Systems*. IEEE Press, 2017, pp. 22–25.

[2] J. Reason and A. Hobbs, *Managing maintenance error: a practical guide*. CRC Press, 2017.

[3] S. Dekker, *Just culture: restoring trust and accountability in your organization*. CRC Press, 2018.

[4] F. H. Hawkins, *Human Factors in Flight*. Gower Technical Press, 1987.

[5] Maritime and Coastguard Agency, "Improving Safety and Organisational Performance Through a Just Culture," 2014. [Online]. Available: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/286139/just_culture.pdf [Accessed: Dec-18]

[6] B. Misiurek, *Standardized Work with TWI: Eliminating Human Errors in Production and Service Processes*. CRC Press, 2016.

[7] B. Krebs. (2016) DDoS on Dyn Impacts Twitter, Spotify, Reddit. [Online]. Available: https://krebsonsecurity.com/2016/10/ddos-on-dyn-impacts-twitter-spotify-reddit/ [Accessed: Dec-18]

[8] G. Martin, S. Ghafur, J. Kinross, C. Hankin, and A. Darzi, "Wannacry a year on." *BMJ (Clinical research ed.)*, vol. 361, p. k2381, 2018.

[9] F. Nietzsche, *Götzen-Dämmerung, oder, Wie man mit dem Hammer philosophirt*. C.G. Naumann, 1889.

[10] S. Dekker, *Just culture: balancing safety and accountability, 2nd Ed.* CRC Press, 2018.

[11] Y. Cohen-Charash and P. E. Spector, "The role of justice in organizations: A meta-analysis," *Organizational behavior and human decision processes*, vol. 86, no. 2, pp. 278–321, 2001.

[12] J. Allspaw. (2012) Blameless PostMortems and a Just Culture. [Online]. Available: https://codeascraft.com/2012/05/22/blameless-postmortems/ [Accessed: Jan-17]

[13] P. Taylor, S. Allpress, M. Carr, E. Lupu, and J. Norton et al, *Internet of Things: Realising the Potential of a Trusted Smart World*. Royal Academy of Engineering, 2018.

[14] V. F. Mancuso, "Human Factors in Cyber Warfare II," *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 58, no. 1, pp. 415–418, 2014.

[15] R. W. Proctor and J. Chen, "The role of human factors///ergonomics in the science of security: decision making and action selection in cyberspace," *Human factors*, vol. 57, no. 5, pp. 721–727, 2015.

[16] P. Carayon, "Human factors of complex sociotechnical systems," *Applied ergonomics*, vol. 37, no. 4, pp. 525–535, 2006.

[17] R. Fielding. (2017) Miss IG Geek: Just Culture 2: Risky Behaviour. [Online]. Available: http://missinfogeek.net/just-culture-2-risky-behaviour/ [Accessed: Dec-18]

[18] R. Mogull. (2018) The Security Profession Needs to Adopt Just Culture. [Online]. Available: https://securosis.com/blog/the-security-profession-needs-to-adopt-just-culture [Accessed: Jun-18]

[19] T. Hayashi, "Schemes for realizing total security in information systems," in *Proc. of 5th Int. Conf. on ICT and Higher Education*, 2006.

[20] M. Bromiley. (2015) Human factors in clinical practice. [Online]. Available: https://vimeo.com/177542101 [Accessed: Jan-17]

[21] J. Reason, "Managing the risks of organizational accidents," 1997.

[22] H. Somerville and P. May. (2014) Use of illicit drugs becomes part of Silicon Valley's work culture. [Online]. Available: https://www.mercurynews.com/2014/07/25/use-of-illicit-drugs-becomes-part-of-silicon-valleys-work-culture/ [Accessed: Aug-18]

[23] B. Blume, "Alcohol and drug abuse in the encyclopedia of occupational health and safety," *International Labour Office,*, pp. 1572–1577, 1998.